


#10A
DR. J. L. LEE
10-4-04

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:  Woobin Lee
Title: CIRCUIT AND METHOD FOR PERFORMING A TWO-DIMENSIONAL TRANSFORM DURING THE PROCESSING OF AN IMAGE

Serial No.: 09/782,509

Filing Date: February 12, 2001

Examiner/Unit: Duy M. Dang / 2621

Attorney Docket No.: 1552-7-10

RECEIVED

AUG 0 9 2004

Technology Center 2600

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited in the United States Postal Service as First Class mail in an envelope addressed to: MS FEE - AMENDMENT, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this 29th day of July, 2004.


Signature

AMENDMENT AND RESPONSE UNDER 37 CFR §1.111

July 29, 2004

TO THE ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231:

In response to the most recent Office Action in this case mailed on March 29, 2004, the Applicant's attorney requests amendment of the above-referenced application as follows.

08/03/2004 JADD01 00000001 09782509

08/03/2004
02 FC:2201

55.00 OP
258.00 OP

I. AMENDMENT

In the Description:

Please amend the paragraph beginning at page 7, line 22 as follows:

The 2-D DCT $F(v, u)$ is given by the following equation:

1)
$$F(v, u) = \frac{2}{N} C(v) C(u) \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} f(y, x) \cos\left(\frac{(2[y+1])v\pi}{[(16)]2N}\right) \cos\left(\frac{(2[x+1])u\pi}{[(16)]2N}\right)$$

$$C(v) = \frac{1}{\sqrt{2}} \text{ for } v = 0, C(v) = 1 \text{ otherwise}$$

$$C(u) = \frac{1}{\sqrt{2}} \text{ for } u = 0, C(u) = 1 \text{ otherwise}$$

where v is the row and u is the column of the corresponding transform block, and N is the dimension of v and u ($N = 8$ for an 8×8 block). For example, if $F(v, u)$ represents the block 37 (Figure 3) of transform values, then $F(1, 3) = D_{13}$. Likewise, $f(y, x)$ is the pixel value in row y , column x of the corresponding pre-compression block. For example, if $f(y, x)$ represents the block 22a (Figure 1B) of pre-compression luminance values, then $f(0, 0) = Y_{(0,0)A}$. Thus, each transform value $F(v, u)$ depends on all of the pixel values $f(y, x)$ in the corresponding pre-compression block.

Please amend the paragraph beginning at page 19, line 16 as follows:

Figure 15 illustrates a map operation that the cluster 114a of Figure 13 can execute according to an embodiment of the invention. For example, a source register Reg0 is divided into eight 8-bit partitions 0-7 and contains the data that the cluster 114a is to map into a destination register Reg1, which is also divided into eight 8-bit partitions 0-7. A 32-bit partition of a control register Reg2 (only one 32-bit partition shown for clarity) is divided into eight 4-bit partitions 0-7 and contains identification values that control the mapping of the data from the source register Reg0 to the destination register Reg1. Specifically, each partition of the control register Reg2 corresponds to a respective partition of the destination

AD
CD

register Reg1 and includes a respective identification value that identifies the partition of the source register Reg0 from which the respective partition of the destination register Reg1 is to receive data. For example, the partition 0 of the control register Reg2 corresponds to the partition 0 of the destination register Reg1 and contains an identifier value "2". Therefore, the cluster 114a loads the contents of the partition 2 of the source register Reg0 into the partition 0 of the destination register Reg1 as indicated by the respective pointer between these two partitions. Likewise, the partition 1 of the control register Reg2 corresponds to the partition 1 of the destination register Reg1 and contains the identifier value "5". Therefore, the cluster 114a loads the contents of the partition 5 of the source register Reg0 into the partition 1 of the destination register Reg1. The cluster 114a can also load the contents of one of the source partitions into multiple destination partitions. For example, the partitions 3 and 4 of the control register Reg2 both include the identification value "6". Therefore, the cluster 114a loads the contents of the partition 6 of the source register Reg0 into the partitions 3 and 4 of the destination register Reg1. In addition, the cluster 114a may not load the contents of a source partition into any of the destination partitions. For example, none of the partitions of the control register Reg1 contains the identity value "7". Thus, the cluster 114a does not load the contents of the partition 7 of the source register Reg0 into a partition of the destination register Reg1.

Please amend the paragraph beginning at page 20, line 15 as follows:

AB

Figure 16 illustrates a 4-point-vector-product operation that the cluster 114a (Figure 13) can execute according to an embodiment of the invention. The cluster 114a loads two 4-point vectors from the register file 120a into the PLC register 136a and two 4-point vectors into the register PLV 138a, where each vector value is 16 bits. For example, during a first clock cycle, the cluster 114a loads the even-odd separated first row of transform values D_{00} , D_{02} , D_{04} , D_{06} , D_{01} , D_{03} , D_{05} , and D_{07} in the block 37 (Figure 3) into the PLC register 136a as shown. During a second clock cycle, the cluster 114a loads the first row of Masaki's four 16-bit even constants (equation (12)) and the first row of Masaki's four 16-bit odd constants into the PLV register 138a as shown. During a third clock cycle, the IFG-unit 118a multiplies the contents of each corresponding pair of partitions of the

registers 136a and 138a, adds the respective products, and loads the results into a 32-bit partition of Reg0 (only one 32-bit partition shown for clarity). That is, the unit 118a multiplies D_{00} by M_{e3} , D_{02} by M_{e2} , D_{04} by M_{e1} , D_{06} by M_{e0} , D_{01} by M_{o3} , D_{03} by M_{o2} , D_{05} by M_{o1} , and D_{07} by M_{o0} , sums the products $D_{00} \times M_{e3}$, $D_{02} \times M_{e2}$, $D_{04} \times M_{e1}$, and $D_{06} \times M_{e0}$ to generate the even Masaki value de_{00} , sums the products $D_{01} \times M_{o3}$, $D_{03} \times M_{o2}$, $D_{05} \times M_{o1}$, and $D_{07} \times M_{o0}$ to generate the odd Masaki value do_{00} , and loads de_{00} and do_{00} into respective halves of the 32-bit partition of Reg0. As discussed below in conjunction with Figures 17 and 18, the unit 118a can use the pair-wise add and subtract and the divided-by-two operations (Figures 14A – 14B) on the Reg0 to generate the intermediate inverse-transform values l'_{00} and l'_{07} of equation (13).

Please amend the paragraph beginning at page 22, line 11 as follows:

Figure 17 illustrates an implicit block transpose that the computing unit 112 performs according to an embodiment of the invention. As discussed above, this implicit transpose allows the unit 112 to generate the transposed block 86 (Figure 9) of values l' directly from the pair-wise add and subtract and the divide-by-two operations (equations (13) and (14)).

The brackets represent 64-bit registers of the register file 120a, and the parentheses represent respective 32-bit partitions of these registers. Furthermore, the dual subscripts of the Masaki values indicate their position within their own row and identify the row of transform values D from which they were generated. For example, de_{00} is the first even Masaki value in the row of Masaki values, i.e., QD_e , that were generated from the first row of transform values $D_{00} - D_{07}$ of the block 37 (Figure 3). Similarly, de_{10} is the first even Masaki value in the row of Masaki values that were generated from the second row of transform values $D_{10} - D_{17}$ of the block 37.